

# Limbaje Formale și Automate

## Tema 3 - Regex parser

Termen de predare: **17 Ianuarie 2021 - 23:55**

Responsabili temă:  
Alex-Bogdan ANDREI  
Mihai-Valentin DUMITRU

Data publicării: *17 Decembrie 2020*  
Ultima actualizare: *05 Ianuarie 2021*

# 1 Cerință

În continuarea temei 2, dorim să transformăm o Expresie Regulată (ER) într-un Automat Finit Determinist (AFD). Pentru a face asta, expresia regulată, prezentată ca sir, trebuie **parsată**, iar arborele de parsare rezultat va fi folosit pentru a genera AFN-ul echivalent. Pentru parsare și construirea arborelui de parsare veți folosi (implementa) un Automat Push-Down (APD).

Ca pași sugerați pentru tema:

1. Pentru reprezentarea expresiilor regulate, veți implementa Tipul de Date Abstract (TDA-ul) prezentat la curs, în Python. Varianta recomandată de implementare este o ierarhie de clase.
2. PDA-ul va parsă expresia regulată și va crea o instanță de ER
3. Aplicați algoritmul de transformare ER în AFN
4. Folosiți algoritmul implementat în *Tema 2* pentru a transforma AFN-ul în AFD.

Tema va fi împartită în 2 subtask-uri:

- **(1p)** Transformarea ER - AFN
- **(0.5p)** Transformarea AFN - AFD

Astfel, în cazul în care tema 2 nu a fost complet implementată (sau deloc), poate fi încercată în a 2-a parte a acestei teme.

## 2 Format input

Veți primi ca input un string ce reprezintă o expresie regulată. Alfabetul va trebui construit de voi în funcție de literele folosite. Acesta diferă de la test la test. Expresia regulată este compusă din:

- litere mici din intervalul 'a'...'z'
- $e_1e_2$  sau  $(e_1e_2)$  - concatenarea a 2 subexpresii
- $e_1|e_2$  sau  $(e_1|e_2)$  - reuniunea a 2 subexpresii
- $(e_1)^*$  - kleene star aplicat unei subexpresii

Exemple:

```
((bb)*(ab)*)
```

```
((bb)*)*|(((b)|(a))|((b)|(a)))
```

Va trebui sa tineti cont de paranteze ( ), acestea schimbând prioritatea operatiilor.

Exemplu:

```
(aa)* vs aa*
```

### 3 Format output

#### Task 1 - Transformare ER - AFN

Va trebui să formatați AFN-ul rezultat din conversie cu formatul întâlnit și la [laboratorul 2](#):

- pe prima linie, un întreg ce reprezintă numărul de stări
- starea inițială trebuie să fie 0
- pe a doua linie, lista de stări finale, separate de câte un spațiu
- pe următoarele linii, până la sfârșitul inputului, câte o tranziție, constând într-o stare, un simbol, apoi o listă de stări următoare - elementele fiind separate de câte un spațiu

Exemplu de output NFA:

```
3
2
0 a 0
0 b 0 1
1 a 2
1 b 2
1 eps 2
```

#### Task 2 - Transformare AFN - AFD

*Output-ul va fi acelasi din tema 2.*

Va trebui să formatați AFD-ul rezultat din conversie cu formatul întâlnit și la [laboratorul 2](#):

- pe prima linie, un întreg ce reprezintă numărul de stări

- starea inițială trebuie să fie 0
- pe a doua linie, lista de stări finale, separate de câte un spațiu
- pe următoarele linii, până la sfârșitul inputului, câte o tranziție, constând într-o stare, un simbol și o stare următoare - elementele fiind separate de câte un spațiu

Exemplu de output DFA:

```
4
2 3
0 b 1
0 a 0
1 b 2
1 a 3
2 b 2
2 a 3
3 b 1
3 a 0
```

## 4 Rulare

Va trebui să aveți un fișier `main.py`, care va fi rulat astfel:

```
$ python3 main.py <input-file> <output-file1> <output-file2>
```

Unde:

- `<input-file>` este calea către fișierul de input ce conține descrierea unei ER
- `<output-file>1` este calea către primul fișier de output ce conține descrierea AFN-ului rezultat din aplicarea algoritmului de conversie
- `<output-file>2` este calea către al doilea fișier de output ce conține descrierea AFD-ului rezultat din aplicarea algoritmului de conversie

În rest, sunteți liberi să structurați codul cum vreți.

## 5 Testare

*TODO: Veți avea la dispoziție toate testele + ref-uri și checker local până pe 19 decembrie*

## 6 Trimitere

*TODO: Tema va fi incarcata pe vmchecker2. Enuntul va fi actualizat in momentul in care instanta pentru tema 3 va fi creata până pe 20 decembrie*

Submisia trebuie să conștea într-o arhivă zip care trebuie să conțină cel puțin un fișier `main.py` direct în arhivă (nu într-un alt subdirector!). În rest, puteți include orice alte fișiere de cod necesare pentru funcționarea temei. Arhiva trebuie să conțină și un fișier README în care să descrieți pe larg implementarea voastră.

Numele arhivei trebuie să fie de forma: `Prenume_Nume_grupa_lfa_tema3.zip`.

Exemplu: `Ion_Andrei_Popescu_333CB_lfa_tema3.zip`.